

## Midterm 1 Coding Questions

Deadline: Check Moodle

### Problem 1

You are tasked with implementing a program that allows a user to dynamically sort a list of integers based on their preferred sorting order. The user should be able to specify whether the sorting should be in ascending or descending order.


You are provided with a skeleton file, you are not allowed to modify any other part of code except **insertion\_sort()** function from line 16 in **main1.py**.

```
1  #012345
2  import sys
3
4  def insertion_sort(arr, ascending=True):
5      """
6      Perform Insertion Sort on the given list.
7
8      Parameters:
9      - arr (list): The list to be sorted.
10     - ascending (bool): If True, sort in ascending order; otherwise, sort in descending order.
11
12     Returns:
13     - list: The sorted list.
14     """
15     #WRITE YOUR CODE HERE
16
17     #DO NOT WRITE ANYTHING BELOW THIS
18     return arr
19
20
21 def main():
22     # Check for the correct number of command line arguments
23     if len(sys.argv) != 2:
24         print("Usage: python script.py [A/D]")
25         sys.exit(1)
26
```

### Requirements

#### 1. Input

- ◆ The skeleton file contains an initialized list of 10 numbers.
- ◆ A command line argument is used to specify the sorting order e.g 'A' for ascending and 'D' for descending.

input

Command line arguments:

Standard Input: ☒ Interactive Console ☐ Text

**2. Order Preference**

- ◆ Allow the user to specify whether the sorting should be in ascending or descending order.

**3. Sorting**

- ◆ Implement the simple insertion sort algorithm to sort the list based on the user's preferences.
- ◆ Example: Let first element as minimum or maximum, loop through the list and compare it with next element.

**4. Output**

- ◆ The program should only print the sorted list which is already defined in the skeleton file, you are not allowed to write any print statement in the code.

**5. Save the File Offline**

- ◆ Click the "Download Code" button/icon to save the file offline.

**6. Submission**

- ◆ Include your Student ID as comment at the top of your code i.e #012345. Rename the main.py file to q1.py.

**7. Grading**

- ◆ (1 point) Correct submission of working code with no errors and loop implementation.
- ◆ (5 point) Correct solution to the problem.

## Problem 2

You are tasked with implementing a program that reverses a given string without using any built-in string reversal functions.

You are provided with a skeleton file, you are not allowed to modify any other part of code except **reverse\_string()** function from line 15 in **main2.py**.

```
1  #012345
2  def reverse_string(string):
3      """
4      Reverse a string without using string functions.
5
6      Parameters:
7      - string (str): The string to be reversed.
8
9      Returns:
10     - str: The reversed string.
11     """
12
13     reversed_string = ""
14     #WRITE YOUR CODE HERE
15
16     #DO NOT WRITE ANYTHING BELOW THIS
17     return reversed_string
18
19 def main():
20     # Default string
21     original_string = "Hello world"
22
```

## Requirements

### 1. Input

- ◆ The skeleton file contains an initialized string.

### 2. String Reversal

- ◆ Implement a function called `reverse_string` that takes a string as input and returns the reversed string without using any built-in string reversal functions.
- ◆ Example: Start with an empty string and iterate through the input string to build the reversed string character by character, starting from the last character and moving towards the first character.

**3. Output**

- ◆ The program should only print the reversed string which is already defined in the skeleton file, you are not allowed to write any print statement in the code.

**4. Save the File Offline**

- ◆ Click the “Download Code” button/icon to save the file offline.

**5. Submission**

- ◆ Include your Student ID as comment at the top of your code i.e #012345. Rename the main.py file to q2.py.

**6. Grading**

- ◆ (1 point) Correct submission of working code with no errors and loop implementation.
- ◆ (5 point) Correct solution to the problem.

### Problem 3

You are tasked with implementing a program that counts the occurrences of a substring within a given string without using any built-in string functions..

You are provided with a skeleton file, you are not allowed to modify any other part of code except **count\_substring\_occurrences()** function from line 15 in **main3.py**.

```
1  #012345
2  def count_substring_occurrences(string, substring):
3      """
4      Count the occurrences of a substring in a given string.
5
6      Parameters:
7      - string (str): The main string to search within.
8      - substring (str): The substring to count occurrences of.
9
10     Returns:
11     - int: The number of occurrences of the substring in the string.
12     """
13     count = 0
14     #WRITE YOUR CODE HERE
15     |
16     #DO NOT WRITE ANYTHING BELOW THIS
17     return count
18
19  def main():
20     # Provided string and substring
21     original_string = "hello, hello, hello world!"
22     original_substring = "hello"
23
```

### Requirements

#### 1. Input

- ◆ The skeleton file contains an initialized string and initialized substring.

#### 2. String Reversal

- ◆ Implement a function called `count_substring_occurrences` that takes a string and a substring as input and returns the count of occurrences of the substring within the string. You should not use any built-in string functions.
- ◆ Example: you can start with an initial count of 0. Then, iterate through the main string character by character. At each character position, compare the current slice

of the string with the given substring. If they match, increment the count. Continue this process until you've iterated through the entire string.

### 3. Output

- ◆ The program should only print the count of occurrences of the substring, which is already defined in the skeleton file. Do not include any print statements in the code.

### 4. Save the File Offline

- ◆ Click the "Download Code" button/icon to save the file offline.

### 5. Submission

- ◆ Include your Student ID as comment at the top of your code i.e #012345. Rename the main.py file to 3.py.

### 6. Grading

- ◆ (1 point) Correct submission of working code with no errors and loop implementation.
- ◆ (5 point) Correct solution to the problem.

## Problem 4

You are tasked with implementing a program that filters prime numbers from a sorted list. You are allowed to use functions from the math library if needed.

You are provided with a skeleton file, you are not allowed to modify any other part of code except `filter_prime_numbers()` function from line 17 in `main4.py`.

```
1  #012345
2  import sys
3  import math
4
5  def filter_prime_numbers(arr):
6      """
7      Filter prime numbers from the given list.
8
9      Parameters:
10     - arr (list): The list to be filtered.
11
12     Returns:
13     - list: The list containing only prime numbers from the original list.
14     """
15     prime_numbers = []
16     #WRITE YOUR CODE HERE
17
18     #DO NOT WRITE ANYTHING BELOW THIS
19     return prime_numbers
20
21 def main():
22     # Initialize a sorted list of integers for testing
23     original_list = [2, 4, 5, 9, 10, 13, 14, 15, 17, 21]
24
25
```

## Requirements

### 1. Input

- ◆ The skeleton file contains an initialized list of 10 integers.

### 2. Prime Number Filtering

- ◆ Implement a function called `filter_prime_numbers()` that takes a sorted list of integers as input and returns a list containing only the prime numbers from the original list.

- ◆ Example: Iterate through each number in the sorted list. For each number, check if it is divisible by any number from 2 to the square root of the number. If it is not divisible by any number in this range, it is a prime number and should be included in the output list.

### 3. Output

- ◆ The program should only print the list of prime numbers, which is already defined in the skeleton file. Do not include any print statements in the code.

### 4. Save the File Offline

- ◆ Click the “Download Code” button/icon to save the file offline.

### 5. Submission

- ◆ Include your Student ID as comment at the top of your code i.e #012345. Rename the main.py file to lab4.py.

### 6. Grading

- ◆ (1 point) Correct submission of working code with no errors and loop implementation.
- ◆ (5 point) Correct solution to the problem.

### Submission Instructions for This Midterm

- ◆ Place q1.py, q2.py, q3.py, and q4.py in a folder and compress to a zip file. Submit it to moodle.